ITRI 625

Pfleeger Chapter 7

Database security requirements

- **Physical database integrity**. The data in a database should be immune to physical problems, such as power failures. Someone should be able to reconstruct it if it is destroyed in a catastrophe.

- **Logical database integrity**. The structure of the database should be preserved. For example a modification in one field should not affect the other fields.

- **Element integrity**. The data contained in each element should be accurate. Can be ensured in three ways:

1. Field checks, where data type and format is checked.
2. Access control, where only certain people can make modifications.
3. Change log, where a log is kept of all changes made, thereby simplifying corrections.

Security requirements *cont.*

- **Auditability**. A record that is kept of all accesses that were made to the database, thereby assisting with the database's integrity.

- **Access control**. The database administrator specifies who should be allowed access to which data, at the view, relation, field, record or even element level.

- **User authentication**. The DBMS usually performs its own authentication, which in turn improves security and integrity.

- **Availability.** The DBMS tends to be taken for granted, and quite often "unavailability" is merely the system servicing another user.

Update integrity

- A system failure in the middle of modifying data is a serious problem (see page 514).

- The two-phase update has been developed to address this problem.

1. The intent phase. The DBMS gathers the resources it needs to complete the update, it prepares everything, but makes no changes. It is a repeatable phase and therefore no harm is done if a system failure occurs during this phase. The last event is committing, during which a commit flag is written. After committing the DBMS starts making permanent changes.

2. Also a repeatable phase during which the changes are made. If a failure occurs, the system can repair it by repeating the phase. Once this phase is complete, the database is complete.

Inference

- A way to infer or derive sensitive data from non-sensitive data.
- Direct attacks. Where users try to determine values of sensitive fields by seeking them directly with queries that yield few records.
- Indirect attacks (see page 521). Seeks to infer a final result based on one or more intermediate statistical results. Only neutral data is released, identifying characteristics are removed.
- Tracker attacks (see page 524). A DBMS may conceal data when a small number of entries make up a large proportion of the data revealed. These attacks fool the manager into locating the desired data by using additional queries that produce small results.

Inference *cont.*

- Linear system vulnerability (see page 525). With a little logic algebra, and luck in the distribution of the database contents, it may be possible to construct a series of queries that returns results relating to several different sets.

- See pages 529 – 535 for controls for inference attacks.

- There are no perfect solutions to the inference problem. The three main approaches to controlling it are:

1. Suppress obviously sensitive information.
2. Track what the user knows.
3. Disguise the data.

Multilevel databases

- Data cannot always only be classified as sensitive or non-sensitive.
- Three characteristics of database security that emerge are:

1. The security of a single element may be different from the security of other elements of the same record or from other values of the same attribute.
2. Two levels are inadequate to represent some security situations.
3. The security of an aggregate may differ from the security of the individual elements.

Proposals for multilevel security

- **Partitioning.** Where the database is divided into separate databases each at its own level of sensitivity. It does however destroy the advantage of elimination of redundancy and improved accuracy.
- **Encryption**. Each level of sensitive data can be stored in a table encrypted under a key unique to the level of sensitivity.
- **Integrity lock**. Provides both integrity and limited access. Each data item consists of three pieces. The actual data item itself, a sensitivity label and a checksum. The checksum is computed to prevent unauthorized modification.
- **Sensitivity lock.** A combination of a unique identifier and the sensitivity level. Because the identifier is unique, each lock relates to one particular record.

Security in data mining

- In a largely automated way, data mining applications sort and search through data.
- They present probable relationships, but these are not necessarily cause-and-effect relationships.
- **Privacy and sensitivity**. Although summarized results are used, individual privacy can still suffer. It suffers from the same kinds of aggregation and inference found in databases.
- **Data correctness and integrity.** Connecting the dots refers to drawing conclusions from relationships between discrete bits of data. The correct data should be collected correctly.

Security in data mining *cont.*

- **Correcting mistakes in data.** Quite often correcting a mistake requires a mistake to be rectified on a master record. In typical marketing scenarios this tends not to be the case. Data mining often takes place across databases that do not have shared keys, so they use data fields as keys. This typically leads to a mistake occurring in several places.

- **Using comparable data.** Data semantics is very important in data mining. When comparing fields in two different databases, they should be in the same format, to avoid badly distorted statistics.

- **Eliminating false matches.** Data mining will inevitably produce false positives and missed connections. We need to be sensitive to the inherent inaccuracy of data mining approaches. Correctness of results and interpretations are major security issues.

Questions

- Any questions?